BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

⑲日本国特許庁（ＪＰ）　⑪特許出願公開

⑫公開特許公報（Ａ）　昭62－283497

⑤Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　昭和62年(1987)12月9日
Ｇ 11 Ｃ　17/00	３０７	6549－5Ｂ	

審査請求　未請求　発明の数　1　（全8頁）

㊹発明の名称　プログラマブルリードオンリメモリの書き込み回数管理方式

㉑特　願　昭61－124732

㉒出　願　昭61(1986)5月31日

㉗発明者　仲田　眞一　東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

㉑出願人　キヤノン株式会社　東京都大田区下丸子３丁目30番２号

㉔代理人　弁理士　小林　将高

明　細　書

１．発明の名称

プログラマブルリードオンリメモリの書き込み回数管理方式

２．特許請求の範囲

記憶領域に書き込まれた情報を電気的に消去可能なプログラマブルリードオンリメモリにおいて、前記記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させることを特徴とするプログラマブルリードオンリメモリの書き込み回数管理方式。

３．発明の詳細な説明

〔産業上の利用分野〕

この発明は、電気的消去可能なプログラマブルリードオンリメモリの書き込み回数の管理方式に関するものである。

〔従来の技術〕

従来のＥＥＰＲＯＭ（Electrical Eresable and Programmable ROM）は、容量も少なく、また書き込むために必要な外部回路が多かった。さらに、チップ内のすべてのデータを消去するモードしか有していなかった。最近は、容量も大きくなるとともに、外部回路も殆ど必要なくＣＰＵのアドレスバス，データバスに結線できるようになり、またＥＥＰＲＯＭ内の１バイトのデータのみの消去も可能となってきた。以上の改良により、使用目的によっては、従来のランダムアクセスメモリ（ＲＡＭ）で構成していた機能の置換が可能となった。

例えば、従来の小型パソコン，日本語ワープロで作成したプログラムや文章，外字等を保存しておくためにメモリカードと云うものがある。これは、必要なときにパソコン，日本語ワープロ等の本体に差し込んでプログラムや文章を記憶させ、本体から引き抜いても、そのデータを記憶しているように、メモリカード内にはＲＡＭと電池が搭載されていた。そこで、メモリカードをＥＥＰＲＯＭで構成することにより、電池を無くすること

ができると考えられた。

〔発明が解決しようとする問題点〕

ところが、ＥＥＰＲＯＭでは従来のＲＡＭのように自由に何度も書き換えられない制約があり、すなわち、あらかじめ設定される書き込み回数を越えて、メモリカードへの書き込みを行なうことにより、記憶しているはずのデータを消失させてしまう等の問題点があった。またＥＥＰＲＯＭに書き込まれたデータのうち、頻繁に書き換えられるデータと書き換え頻度の少ないデータとが存在し、書き換え頻度の高いデータの書き換え回数が所定値を越えると、ＥＥＰＲＯＭへの書き換えが可能にも関わらず書き換え不能となる問題点があった。

この発明は、上記の問題点を解消するためになされたもので、ＥＥＰＲＯＭに書き込まれるデータの消失を防止するとともに、ＥＥＰＲＯＭへの書き込み回数を平均化させるとともに、ＥＥＰＲＯＭ上の書き換え頻度を平均化して、ＥＥＰＲＯＭへの書き換え寿命を延命できるプログラマブルリードオンリメモリの書き込み回数管理方式を得ることを目的とする。

〔問題点を解決するための手段〕

この発明に係るプログラマブルリードオンリメモリの書き込み回数管理方式は、記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させる。

〔作用〕

この発明においては、記憶領域を各ブロック毎に書き込み回数を記憶しておき、この書き込み回数があらかじめ設定される書き込み回数を越えたら、そのブロックへの書き込みを抑止させる。

〔実施例〕

第１図（ａ）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図であり、１はＥＥＰＲＯＭで、例えば書き込み容量が32798 バイト×８ビットで、書き込み回数が１万回に設定してある。ＥＥＰＲＯＭ１は、ポインタブロック１ａおよび予備ポインタブロックＳＰＢ１～ＳＰＢ５０より構成される。ポインタブロック１ａは４アドレス（各１バイト）で構成され、『０～１』番地の２バイトで、書き換え回数ＷＣＮＴ、例えば『1388_{16}』を記憶している。またポインタブロック１ａの『２』番地の１バイトは、ディレクトリＤＢ、例えば『01_{16}』を記憶している。さらに、ポインタブロック１ａの『３』番地の１バイトは、未使用のスタートブロック番号ＯＳＢ、例えば『33_{16}』を記憶している。またポインタブロック１ａの『４』番地の１バイトは、未使用のエンドブロック番号ＯＥＢ、例えば『$8A_{16}$』を記憶している。

第１図（ｂ）はこの発明の装置構成の一例を説明するブロック図であり、１１はＣＰＵで、ＲＯＭ１１ａ，ＲＡＭ１１ｂを有し、ＲＯＭ１１ａに格納された第６図に示すフローに準じたプログラムに応じて各部を制御する。１２は入力手段で、データ書き込み装置１３にセットされるＥＥＰＲＯＭ１へのデータ書き込みおよびデータ消去を指示する。なお、ＣＰＵ１１にはデータの転送を行うアキュムレータＡＣＣ，ＢＣＣを有している。

第２図は第１図（ａ）に示すＥＥＰＲＯＭ１の構造を示す模式図であり、２１はブロック番号であり、例えば１２７個のブロックＢＬＯＣＫ１～ＢＬＯＣＫ１２７に分割されている。各ブロックは、例えば２５６バイトで構成され、先頭の２バイトで、そのブロックが更新された回数、すなわち、後述する更新回数が記憶されている。次に続く２５３バイトは記憶データＤＡＴＡが記憶されており、最後の１バイトは、記憶データＤＡＴＡがこのブロックに留まるか、または他のブロックに及ぶかどうかを示す継続ブロックエリアＣＢがあり、他のブロックに記憶データＤＡＴＡが及ぶ場合は、継続ブロックエリアＣＢには継続するブロック番号が記憶され、他のブロックに記憶データＤＡＴＡが及ばない場合は、継続ブロックエリアＣＢには『FF_{16}』が記憶されている。

第３図は第２図に示す各ディレクトリブロック構造を説明する模式図であり、３０は前記ディレ

クトリＤＢに指示されるディレクトリブロック．３１は前記ディレクトリブロック３０の更新カウンタで．例えば２バイトで構成される．３２はファイル領域で．各ファイル名が１２バイトで記憶される．３３はスタートブロック番号エリア（ＳＢ）で．例えば１バイトで構成され．ファイルのスタートブロック番号が記憶されている．３４はエンドブロック番号エリア（ＥＢ）で．例えば１バイトで構成され．ファイルのエンドブロック番号が記憶されている．３５はチェーンブロックエリア（ＣＢ）で．ディレクトリブロック３０に継続するディレクトリブロックの有無を記憶する．例えばチェーンブロックエリア３５が『ＦＦ$_{16}$』となる．なお．ディレクトリブロック３０は．例えば１８個のファイル領域３２で構成される．

次に第１図（ａ）および第３図を参照しながらＥＥＰＲＯＭ１の構造について説明する．

第１図（ａ）に示すようにポインタブロック１ａの書き換え回数ＷＣＮＴに．例えば『１３８$_{16}$』が記憶されているとすると．５０００回の更新が行われたことを示し．またディレクトリＤＢには『０１$_{16}$』が記憶されているので．ディレクトリＤＢに指示されるディレクトリブロック３０のブロック番号が『１』で．そのディレクトリブロック３０の更新カウンタ３１には．『１４２Ｆ$_{16}$』が記憶されている．これは．このディレクトリブロック３０を５１６７回更新したことを示し．ファイル領域３２のファイル（ＦｉＬｅ）１（ファイル名）はスタートブロック番号エリア３３が『０２$_{16}$』で．エンドブロック番号エリア３４が『０５$_{16}$』となっているため．ブロックＢＬＯＣＫ２から始まり．ブロックＢＬＯＣＫ５で終ることになる．またファイル領域３２のファイル２は．スタートブロック番号エリア３３が『０Ａ$_{16}$』で．エンドブロック番号エリア３４が『０Ｆ$_{16}$』となっているため．ブロックＢＬＯＣＫ１０から始まり．ブロックＢＬＯＣＫ１５で終ることになる．さらに．ファイル領域３２のファイル３（ファイル名）は．スタートブロック番号エリア３３が『１５$_{16}$』で．エンドブロック番号エリア３４が『１８$_{16}$』となっているため．ブロックＢＬＯＣＫ２１から始まり．ブロックＢＬＯＣＫ２４で終ることになる．またファイル領域３２のファイル３の次に『ＦＦ$_{16}$』が書かれているので．このファイル領域３２はファイル３で終了していることになる．

第４図は未使用のＥＥＰＲＯＭ１の状態を説明する模式図であり．第１図（ａ）．第３図と同一のものには同じ符号を付している．

この図から分かるように．未使用のＥＥＰＲＯＭ１のポインタブロック１ａの書き換え回数ＷＣＮＴが『０００１$_{16}$』．ディレクトリＤＢが『０１$_{16}$』．未使用のスタートブロック番号ＯＳＢが『０２$_{16}$』．未使用のエンドブロック番号ＯＥＢが『７Ａ$_{16}$』がそれぞれポインタブロック１ａの０番地から４番地にそれぞれ記憶されている．これにより．ディレクトリＤＢに指示されるブロックＢＬＯＣＫ１を参照すると．更新カウンタ３１に『０００１$_{16}$』が書き込まれているとともに．ファイル領域３２のファイル１に『ＦＦ$_{16}$』が書き込まれており．さらに．チェーンブロックエリア３５に『ＦＦ$_{16}$』が書き込まれており．ＥＥＰＲＯＭ１が未使用状態であることを示している．

さらに．ポインタブロック１ａのスタートブロック番号ＯＳＢおよびエンドブロック番号ＯＥＢには『０２$_{16}$』．『７Ｆ$_{16}$』がそれぞれ書き込まれている．すなわち．ブロックＢＬＯＣＫ２～１２７には先頭の２バイトに『０００１$_{16}$』が書き込まれ．最終の１バイトに各後続のブロックの継続を示すチェーンブロックエリア３５には．ブロックＢＬＯＣＫ２～１２６に対して『０３～７Ｆ$_{16}$』が書き込まれ．ブロックＢＬＯＣＫ１２７のチェーンブロックエリア３５には『ＦＦ』が書き込まれている．このように．各ブロックＢＬＯＣＫ２～１２７は１つのチェイン構造となる．

次に第３図．第５図（ａ）．（ｂ）を参照しながらＥＥＰＲＯＭ１への書き込み動作を説明する．

第５図（ａ）．（ｂ）はＥＥＰＲＯＭ１への書き込み動作を説明する模式図であり．第１図

（ａ），第３図と同一のものには同じ符号を付している。なお、書き込み直前は、第３図に示す状態であったものとする。

まず、各ブロックＢＬＯＣＫのファイル領域３２の先頭が『００$_{16}$』のところを探し当てる。第３図の場合は、ファイル２とファイル３との間に『００$_{16}$』があり、そこにファイル４という名前を１２バイトで書き込み、ポインタブロック１ａの未使用ブロックのスタートブロック番号ＯＳＢを参照して、スタートブロック番号ＯＳＢの指示するブロックＢＬＯＣＫ、すなわち『５７$_{16}$』の先頭の２バイト情報、すなわち、更新カウンタ３１を『１』インクリメントし、その加算値が、例えば１万回を越えているようであれば、ファイル４のチェーンブロックエリア３５が示すブロックＢＬＯＣＫに対して同様の操作を行い、更新カウンタ３１が１万回以下のブロックＢＬＯＣＫを探し当てて、そのブロックＢＬＯＣＫの番号をポインタブロック１ａのスタートブロック番号ＯＳＢに書き込むとともに、ファイル４のデータをブロックＢＬＯＣＫ８７（２５３バイト）に書き込み、ブロックＢＬＯＣＫ８７に溢れるようであれば、ブロックＢＬＯＣＫ８７のチェーンブロックエリア３５の指示するブロックＢＬＯＣＫの更新カウンタ３１を『１』インクリメントして加算値が、例えば１万回を越えているかどうかを調べ、指示されるブロックＢＬＯＣＫの更新カウンタ３１が１万回を越えるようであれば、更新回数が１万回以下のブロックＢＬＯＣＫを探し当て、そのブロックＢＬＯＣＫの番号を直前に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に書き込む。このようにして、データの書き込みが行われ、更新回数が１万回を越えるブロックＢＬＯＣＫが排除されて行く。そして、書き込みデータがなくなるまで同様の操作を行い、最後に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に記憶されていた内容を新しい未使用のスタートブロック番号ＯＳＢに書き換え、ポインタブロック１ａの書き換え回数ＷＣＮＴを『１』インクリメントして『１３８９$_{16}$』となり、最後にデータを書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５を『ＦＦ$_{16}$』にする。そして、ディレクトリブロック３０の最終ブロック番号を記憶するエンドブロック番号エリア３４に最後のデータを書き込んだブロックＢＬＯＣＫの番号を書き込むとともに、更新カウンタ３１を『１』インクリメントすると、第５図（ｂ）に示されるように、更新カウンタ３１が『１４３０$_{16}$』となり、ファイル４のスタートブロック番号エリア３３が『３３$_{16}$』で、エンドブロック番号エリア３４が『３７$_{16}$』となる。

次に第５図（ａ），（ｂ）を参照しながらＥＥＰＲＯＭ１に書き込まれているファイル１の削除動作について説明する。

ディレクトリブロック３０となるブロックＢＬＯＣＫ１よりファイル１を探し、ファイル領域３２の先頭の２バイトを『００$_{16}$』とする。次いで、ディレクトリブロック３０の更新カウンタ３１を『１』インクリメントし、ファイル１のスタートブロック番号エリア３３とエンドブロック番号エリア３４のデータを参照して、ポインタブロック１ａのエンドブロックＯＥＢが指示するブロックのチェーンブロックエリア３５の内容（削除直前までは『ＦＦ$_{16}$』であった）をスタートブロック番号エリア３３の内容に変更し、このブロックの更新カウンタ３１を『１』インクリメントする。すなわち、未使用ブロックの最後に今削除したファイル４を接続するわけである。このようにして、更新カウンタ３１を進めながら何度もファイルの更新，削除を実行して行くうちに、更新カウンタ３１が１万回に接近する。

次に更新カウンタ３１が１万回に到達した場合のアクセス処理について説明する。

まず、ポインタブロック１ａのスタートブロック番号ＯＳＢの内容が示しているブロックＢＬＯＣＫのチェーンブロックエリア３５の内容を新規のスタートブロック番号ＯＳＢとする。次いで、このブロック直前のディレクトリブロック３０の更新カウンタ３１の情報以外の内容を転送する。そして、ポインタブロック１ａのディレクトリＤ

Bに新規のディレクトリブロック番号を書き込み、ポインタブロック1aの書き換え回数WCNTおよび更新カウンタ31を『1』インクリメントする。

一方、ポインタブロック1aの書き換え回数WCNTは1万回を越えた場合は、予備ポインタブロックSPB1～SPB50のうち一番近い予備ポインタブロックへ書き換え回数WCNTの情報以外のデータを転送し、新規のポインタブロックの書き換え回数WCNT（0000_{16}）を『1』インクリメントして『0001_{16}』に設定する。この場合、破棄されたポインタブロック1aの書き換え回数WCNTは1万回以上となり、新のポインタブロック1aの書き換え回数WCNTは1万回以下となる。このようにして、カウンタブロック30およびポインタブロック1aの書き込み削除を管理する。また削除されたファイルが使用していたブロックは未使用ブロックの一番最後に回される。これは、未使用ブロックの使用回数を平均化するためである。

第6図は第1図（a）に示したEEPROM1のデータ書き込み制御動作を説明するためのフローチャートである。なお、(1)～(18)は各ステップを示す。

まず、ディレクトリブロック30の空エリアを探して、新規のファイル名を書き込む(1)。次いで、未使用のスタートブロック番号OSBをCPU11のアキュムレータACCに記憶させる(2)。アキュムレータACCが指示するブロックの書き換え回数WCNTを＋1更新する(3)。ここで、書き換え回数WCNTが10000を越えたかどうかを判断し(4)、YESならばアキュムレータACCの指示するブロックの継続ブロックエリアCBをアキュムレータACCに記憶し(5)、ステップ(3)に戻り、NOならばディレクトリブロック30のスタートブロック番号エリア（SB）33にアキュムレータACCの内容を書き込む(6)。次いで、アキュムレータACCが指示するブロックのデータエリアにデータを書き込む(7)。ここで、書き込みデータがアキュムレータACCが指示するブロックの容量が235バイトを越えるかどうかを判断し(8)、YESならばアキュムレータACCが指示するブロックの継続ブロックエリアCBをアキュムレータBCCに記憶させる(9)。次いで、アキュムレータBCCが指示するブロックの書き換え回数WCNTを＋1更新する(10)。次いで、書き換え回数WCNTが10000を越えたかどうかを判断し(11)、YESならばアキュムレータBCCの指示するブロックの継続ブロックエリアCBを記憶させ(12)、ステップ(10)に戻り、NOならばアキュムレータACCが指示するブロックの継続ブロックエリアCBにアキュムレータBCCの内容を書き込み(13)、ステップ(7)に戻る。

一方、ステップ(8)の判断でNOの場合は、アキュムレータACCが指示する継続ブロックエリアCBを未使用のスタートブロック番号OSBに書き込む(14)。次いで、ポインタブロック1aの書き換え回数WCNTを＋1更新する(15)。次いで、アキュムレータACCが指示するブロックの継続ブロックエリアCBへ「FF_{16}」を書き込む(16)。そして、ディレクトリブロック30の新ファイル位置のエンドブロック番号エリア34へアキュムレータACCの内容を書き込む(17)。次いで、ディレクトリブロック30の書き換え回数WCNTを更新する(18)。

〔発明の効果〕

以上説明したように、この発明は記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させるようにしたので、EEPROMに書き込まれるデータの消失を未然に防げるとともに、不要になったブロックを未使用ブロックの最後尾に接続するようにしたので、各ブロックの書き込み回数を平均化できる利点を有する。

4．図面の簡単な説明

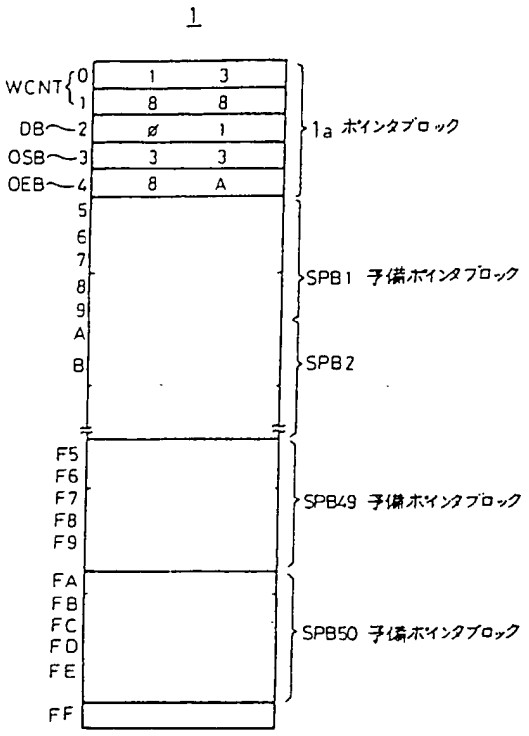
第1図（a）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図、第1図（b）はこの

発明の装置構成の一例を説明するブロック図、第2図は第1図（a）に示すEEPROMの構造を示す模式図、第3図は第2図に示す各ディレクトリブロック構造を説明する模式図、第4図は未使用のEEPROM状態を説明する模式図、第5図（a），（b）はEEPROMへの書き込み動作を説明する模式図、第6図は第1図（a）に示したEEPROMのデータ書き込み動作を説明するためのフローチャートである。

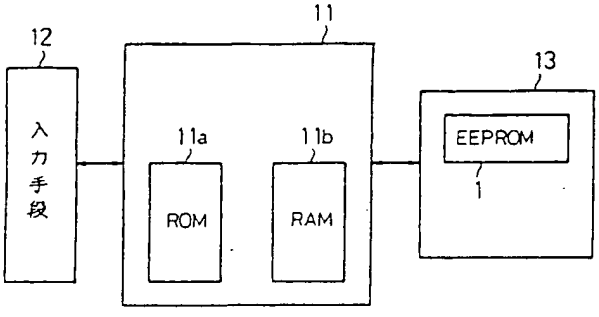
図中、1はEEPROM、1aはポインタブロック、21はブロック番号、30はディレクトリブロック、31は更新カウンタ、32はファイル領域、33はスタートブロック番号エリア、34はエンドブロック番号エリア、35はチェーンブロックエリアである。

代理人　小　林　将　高

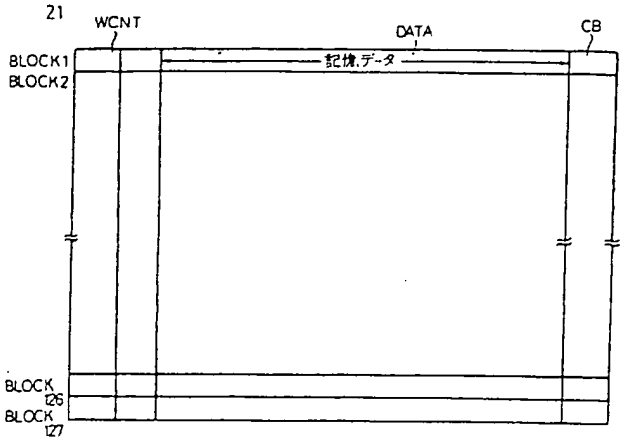
第　1　図　(a)



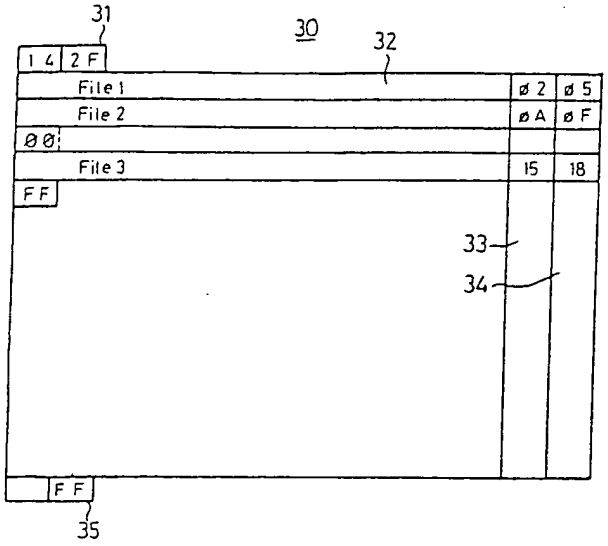
第　1　図　(b)



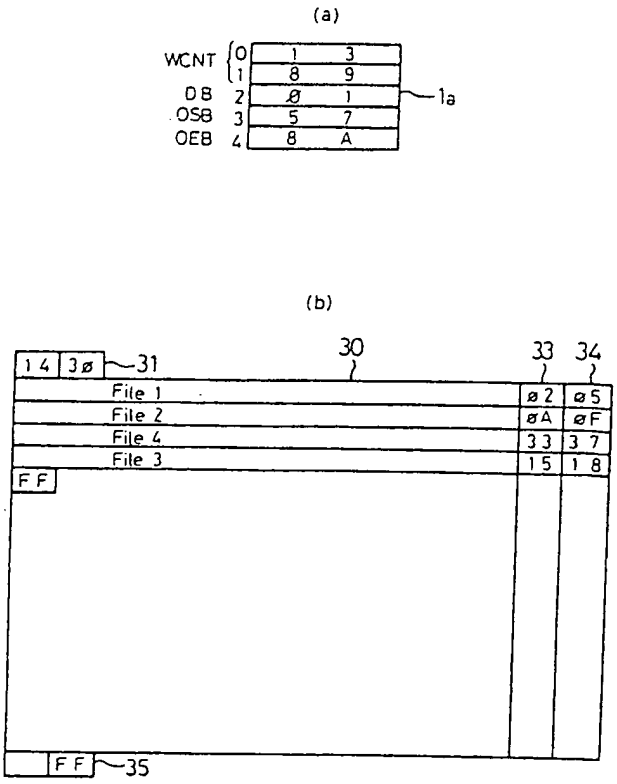
第　2　図



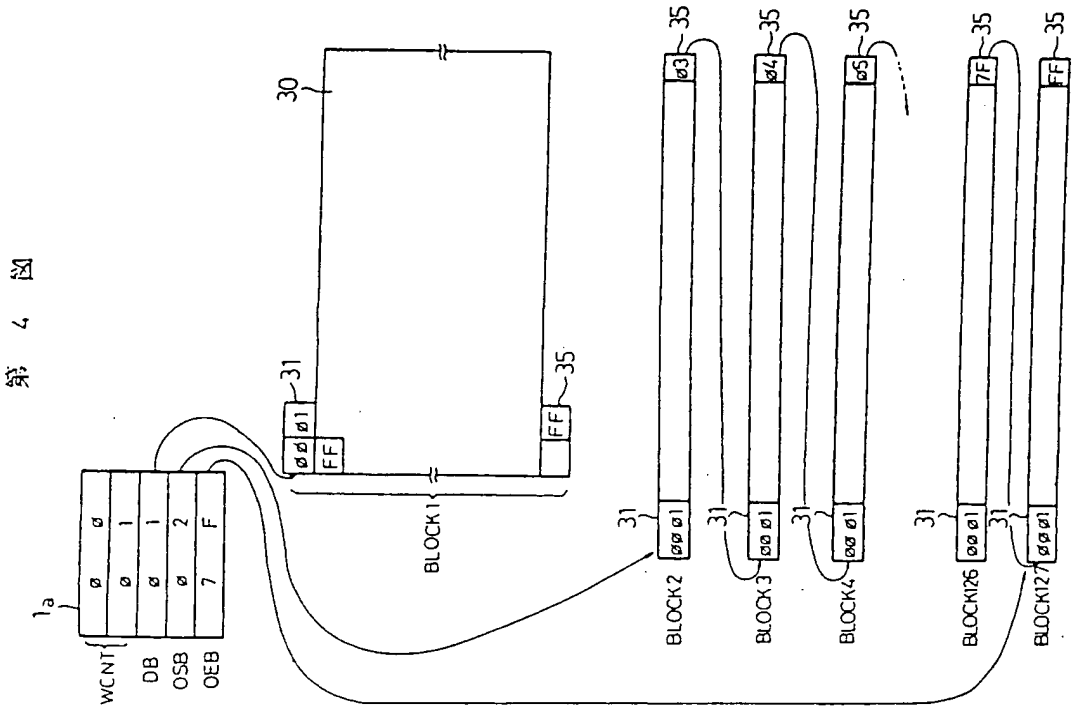
第 3 図


31
30
32
1 4
2 F
File 1
File 2
ø ø
File 3
F F
ø 2
ø 5
ø A
ø F
15
18
33
34
F F
35

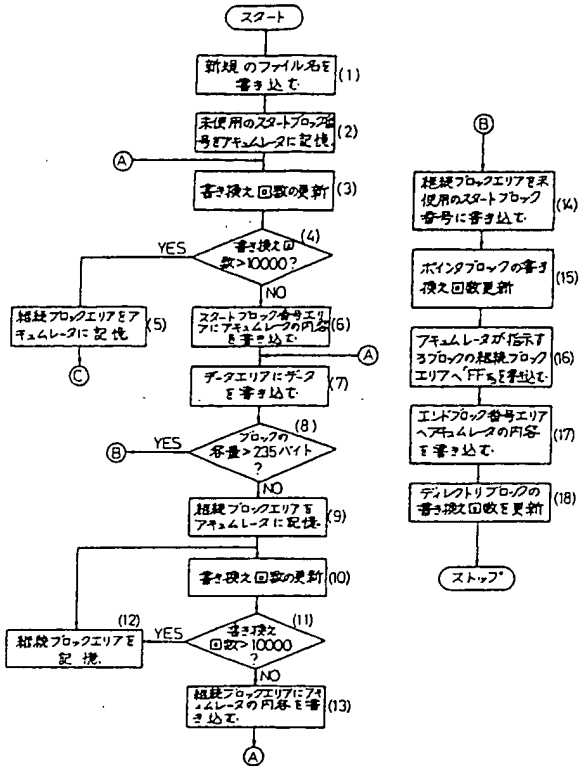
第 5 図


(a)
WCNT
DB
OSB
OEB
0
1
2
3
4
1 3
8 9
ø 1
5 7
8 A
1a
(b)
30
31
33
34
1 4
3 ø
File 1
File 2
File 4
File 3
F F
ø 2
ø 5
ø A
ø F
3 3
3 7
1 5
1 8
35

第 4 図


30
31
35
ø ø 1
F F
BLOCK1
ø 3
ø 4
ø 5
7F
FF
BLOCK2
BLOCK3
BLOCK4
BLOCK126
BLOCK127
1a
WCNT
DB
OSB
OEB

第 6 図


スタート
新規のファイル名を書き込む (1)
未使用のスタートブロック番号をアキュムレータに記憶 (2)
A
書き換え回数の更新 (3)
書き換え回数>10000? (4)
YES
NO
結線ブロックエリアをアキュムレータに記憶 (5)
C
スタートブロック番号エリアにアキュムレータの内容を書き込む (6)
A
データエリアにデータを書き込む (7)
ブロックの容量>235バイト? (8)
B
YES
NO
結線ブロックエリアをアキュムレータに記憶 (9)
書き換え回数の更新 (10)
書き換え回数>10000? (11)
YES
結線ブロックエリアを記憶 (12)
NO
結線ブロックエリアにアキュムレータの内容を書き込む (13)
A
B
結線ブロックエリアを未使用のスタートブロック番号に書き込む (14)
ポインタブロックの書き換え回数更新 (15)
アキュムレータが指示するブロックの結線ブロックエリアへFFを書込む (16)
エンドブロック番号エリアへアキュムレータの内容を書き込む (17)
ディレクトリブロックの書き換え回数を更新 (18)
ストップ

POLYGLOT INTERNATIONAL
Global Management of Language-Related Projects

340 Brannan Street, Fifth Floor
San Francisco, CA 94107 • USA

Tel (415) 512-8800
FAX (415) 512-8982

TRANSLATION FROM JAPANESE

(19) JAPANESE PATENT OFFICE (JP)
(11) Japanese Laid-Open Patent Application (Kokai) No. **62-283497**
(12) Official Gazette for Laid-Open Patent Applications (A)

(51) Int. Cl.[4]:	Classification Symbols:	Internal Office Registration Nos.:
G 11 C 17/00	307	6549-5B

(43) Disclosure Date: December 9, 1987
Request for Examination: Not yet submitted
Number of Inventions: 1
(Total of 8 pages [in original])

---

(54) Title of the Invention: **System for Managing Number of Writes to Programmable Read-only Memory**

(21) Application No. 61-124732
(22) Filing Date: May 31, 1986
(72) Inventor: Shin'ichi Nakada
(71) Applicant: Canon Inc.
(74) Agent: Masataka Kobayashi, Patent Attorney

SPECIFICATION

Title of the Invention

System for Managing Number of Writes to Programmable Read-only Memory

Claims

A system for managing the number of writes to a programmable read-only memory, characterized by the fact that in a programmable read-only memory which stores electrically erasable data that has been written to a memory area, the aforementioned memory area is divided into a plurality of blocks, the number of writes to each block is stored in memory, and when a predetermined number of writes to a block is exceeded, further write operations [to the block] are disabled.

Detailed Description of the Invention

Field of Industrial Utilization

The present invention relates to a system for managing the number of writes to a programmable read-only memory which stores electrically erasable data that has been written to a memory area.

Prior Art

Conventional EEPROM (electrically erasable and programmable ROM) had fairly small capacity and required extensive external circuitry for performing write operations. Furthermore, they did not have a mode that allows all data on a chip to be erased. More recently, capacity has been increased, and it has become possible to connect devices to the CPU address bus or data bus with virtually no need for external circuitry. It has also become possible to erase single bytes of data stored in the EEPROM. These improvements have allowed such devices to replace conventional random-access memory (RAM) for certain functions.

One example is the memory cards used to store programs, documents, foreign-language [documents], and the like created on compact personal computers and Japanese word processors. When required, [the card] is inserted into the console of a personal computer or word processor, and a program, document, or the like is saved to the card; the memory card houses RAM and a battery so that the data is saved even

when the card is removed from the console. By using EEPROM for the memory card, it would be possible to eliminate the battery.

Problems Which the Invention Is Intended to Solve

However, EEPROM has the limitation that, unlike conventional RAM, the number of write operations is not unlimited. Specifically, once a pre-established number of write operations has been exceeded, further writes to the memory card result in the erasure of data previously stored. The data stored in an EEPROM includes both data that is rewritten frequently and data that is rewritten infrequently; once a prescribed number of writes to frequently rewritten data has been reached, further write operations to the EEPROM are disabled, despite the fact that rewriting would be possible.

This invention was developed in order to address the aforementioned drawbacks, and is intended to provide a system for managing the number of writes to a programmable read-only memory which prevents erasure of data that has been written to an EEPROM, and which equalizes the number of writes that can be made to the EEPROM as well as equalizing write frequency to the EEPROM, thereby extending the write life of the EEPROM.

Means Used to Solve the Aforementioned Problems

The system for managing the number of writes to a programmable read-only memory which pertains to the present invention involves dividing the memory area into a plurality of blocks, storing the number of writes to each block, and when a predetermined number of writes to a block is exceeded, disabling further write operations [to the block].

Effect of the Invention

In the present invention, the memory area [is divided into] blocks and the number of writes [to each block] is stored [by the device] so that when the number of write operations [to a block] reaches a preset write count, further writing to the block is disabled.

Practical Examples

Fig. 1(a) is a schematic illustration of a system for managing the number of writes to a programmable read-only memory in a practical example of the present invention. 1 indicates an EEPROM with a write capacity, for example, of

32,798 bytes × 8 bits, set to allow 10,000 write operations. The EEPROM 1 is provided with a pointer block 1a and spare pointer blocks SPB1 through SPB50. The pointer block 1a comprises 4 addressees (one byte each). The two bytes of addresses 0 and 1 contain a write count WCNT, for example, "1388_{16}". The one byte of address 2 of pointer block 1a contains a directory DB, for example, "01_{16}". The one byte of address 3 of pointer block 1a contains an unused start block number OSB, for example, "33_{16}". The one byte of address 4 of pointer block 1a contains an unused end block number OEB, for example, "$8A_{16}$".

Fig. 1(b) is a block diagram illustrating an example of device configuration in the present invention. 11 indicates a CPU which is equipped with ROM 11a and RAM 11b. [The CPU] controls the various components in accordance with a program stored in the ROM 11a and based on the algorithm shown in Fig. 6. 12 indicates an input means which is used to enter data write and data erase commands for the EEPROM 1 located in a data write device 13. The CPU 11 is also equipped with accumulators ACC and BCC for data transfer.

Fig. 2 is a schematic illustration of the configuration of the EEPROM 1 depicted in Fig. 1(a). 21 indicates block numbers assigned to, for example, 127 blocks, BLOCK1 through BLOCK 127. Each block comprises, for example, 256 bytes, with the two lead bytes containing the number of updates, that is, the update count described later. The next 253 bytes contain stored data DATA, and the final byte is a chain block area CB indicating whether the stored data DATA ends with this block or continues to another block. When the stored data DATA continues to another block, the block number of the chained block is held in the chain block area CB; when the stored data DATA does not continue to another block, the chain block area CB contains "FF_{16}".

Fig. 3 is a schematic illustration of the configuration of the directory block depicted in Fig. 2. 30 is the directory block indicated by the aforementioned directory DB. 31 indicates the update counter of the aforementioned directory block 30 and comprises, for example, two bytes. 32 is a file area in which each twelve-byte file name is stored. 33 is a start block number area (SB) comprising, for example, one byte, which holds the file start block number. 34 is an end block number area (EB) comprising, for example, one byte, which holds the file end block number. 35 is a chain block area (CB) which indicates whether there is a directory block that continues on from directory block 30. The chain block area 35 contains, for example, "FF_{16}". The directory block 30 comprises, for example, eighteen file areas 32.

Next, the configuration of the EEPROM 1 will be described referring to Fig. 1(a) and Fig. 3.

As shown in Fig. 1(a), for example, the write count WCNT of the pointer block 1a contains "1388_{16}", indicating that 5,000 updates have been made. Since the directory DB contains "01_{16}", the block number of the directory block 30 indicated by the directory DB is "1". The update counter 31 of the directory block 30 contains "$142F_{16}$", indicating that this directory block 30 has been updated 5,167 times. Since the start block number area 33 for File 1 (file name) in file area 32 contains "02_{16}" and the end block number area 34 contains "05_{16}", the file starts at BLOCK 2 and ends at BLOCK 5. Since the start block number area 33 for File 2 in file area 32 contains "$0A_{16}$" and the end block number area 34 contains "$0F_{16}$", the file starts at BLOCK 10 and ends at BLOCK 15. Since the start block number area 33 for File 3 (file name) in file area 32 contains "15_{16}" and the end block number area 34 contains "18_{16}", the file starts at BLOCK 21 and ends at BLOCK 24. Since "FF_{16}" follows File 3 in file area 32, the file area 32 ends at File 3.

Fig. 4 is a schematic illustration of the EEPROM 1 prior to use. The symbols used are the same as those in Fig. 1(a) and Fig. (3).

As the drawing shows, the following are stored at addresses 0 through 4 in the pointer block 1a: the write count WCNT in the pointer block 1a of the EEPROM 1 is "0001_{16}", the directory DB is "01_{16}", the unused start block number OSB is "02_{16}", and the unused end block number OEB is "$7A_{16}$". In the block BLOCK 1 indicated by the directory DB, the update counter 31 contains "0001_{16}", File 1 in the file area 32 contains "FF_{16}", and the chain block area 35 contains "FF_{14}", indicating that the EEPROM 1 is has not yet been used.

The pointer block 1a start block number OSB and end block number OEB contain "02_{16}" and "$7F_{16}$", respectively. Specifically, the two lead bytes in blocks BLOCK 2 through BLOCK 127 contain "0001_{16}", and the final single bytes (the chain block area 35 which indicates a serial chained block) in each of BLOCK 2 through BLOCK 126 contain "03 - $7F_{16}$". The chain block area 35 of BLOCK 127 contains "FF". Thus, the blocks BLOCK 2 through BLOCK 127 are chained.

Figs. 5(a) and (b) are schematic representations of the write operation to the EEPROM 1. The symbols used are the same as those in Fig. 1(a) and Fig. (3).

First, [the system] searches the file area 32 of each block BLOCK for the leading [characters] "00_{16}". In Fig. 3, "00_{16}" is found between File 2 and File 3, whereupon the 12-byte file name "File 4" is written at this location. Referring to the start block number OSB for the unused block in the pointer block 1a, the data of the

two leading bytes for the block BLOCK indicated by the start block number OSB, specifically, "57_{16}", [is read][1] ; specifically, the update counter 31 is incremented by one, and if the cumulative value exceeds, for example, 10,000, this same operation is performed for the block BLOCK indicated by the chain block area 35 of File 4, and a search is made for a block BLOCK for which [the value in] the update counter 31 is 10,000 or less. [When found,] the number of this block BLOCK is written to the start block number OSB of the pointer block 1a and the data of File 4 is written to BLOCK 87 (253 bytes). If the block BLOCK 87 becomes filled, the update counter 31 of the block BLOCK indicated in the chain block area 35 of the block BLOCK 87 is incremented by one, and a check is made to determine whether the cumulative value exceeds, for example, 10,000; if the update counter 31 of the indicated block BLOCK exceeds 10,000, a search is made for a block BLOCK for which the update count is 10,000 or less. [When found,] the number of this block BLOCK is written to the chain block area 35 of the preceding block BLOCK. In this way, data is written while excluding blocks BLOCK for which the update count exceeds 10,000. This operation is repeated until all the data has been written, whereupon the contents of the chain block area 35 of the last block BLOCK written to replaces the old unused start block number OSB, and the write count WCNT in the pointer block 1a is incremented by one, to "1389_{16}". "FF_{16}" is written to the chain block area 35 of the last block BLOCK to which data was written. The number of the last block BLOCK to which data was written is written to the block number area 34 of the directory block 30 (which stores the final block number), and the update counter 31 is incremented by one. [At the end of this process], the update counter 31 contains "1430_{16}", the start block number area 33 for File 4 contains "33_{16}", and the end block number area 34 "37_{16}", as shown in Fig. 5(b).

The procedure for erasing File 1 which has been written to the EEPROM 1 will be described referring to Figs. 5(a) and (b).

[The system] searches for File 1 from the block BLOCK 1 (the directory block 30) and places "00_{16}" in the two lead bytes of the file area 32. Next, the update counter 31 of the directory block 30 is incremented by one. Referring to the start block number area 33 and end block number area 34 of File 1, the contents of the chain block area 35 of the block indicated by the end block number OEB of the pointer block 1a (which had been "FF_{16}" prior to the erase operation) are changed to reflect the contents of the start block number area 33, and the update counter 31 for this block 30 is

[1] [Translator's note: Uncertain interpolation made necessary by the fact that the author has neglected to provide a verb.]

incremented by one. In other words, File 4 [sic] which has just been erased is attached to the end of the unused block. In this way, the update counter 31 approaches 10,000 as files are repeatedly updated and deleted while advancing the update counter 31.

The access process performed when the update counter 31 has reached 10,000 will now be described.

First, the contents of the chain block area 35 of the block BLOCK indicated by the contents of the start block number OSB of the pointer block 1a are designated as the new start block number OSB. Next, the contents of the directory block 30 preceding this block, with the exception of the update counter 31, are transferred. The new directory block number is written to the directory DB of the pointer block 1a, and the write count WCNT of the pointer block 1a and the update counter 31 are incremented by one.

In the event that the write count WCNT of the pointer block 1a exceeds 10,000, data other than the write count WCNT data is transferred to the closest spare pointer block among the spare pointer blocks SPB1 through SPB50, and the write count WCNT (0000_{16}) of the new pointer block is incremented by one, to (0001_{16}). At this point the write count WCNT of the discarded pointer block 1a exceeds 10,000 and the write count WCNT of the new pointer block 1a is less than 10,000. In this way, write [and] erase [operations] to the counter block 30 [sic] and the pointer block 1a are managed. Blocks used by erased files are returned to the very end of the unused block. This equalizes the number of times unused blocks are used.

Fig. 6 is a flow chart describing the control procedure for data write operations to the EEPROM 1. (1) through (18) indicate steps.

First, [the system] searches for a free area in the directory block 30 and writes a new file name (1). The unused start block number OSB is then placed in the accumulator ACC of the CPU 11 (2). The write count WCNT for the block indicated by the accumulator ACC is incremented by one (3). At this point, it is ascertained whether the write count WCNT exceeds 10,000 (4). If the answer is YES, the chain block area CB of the block indicated by the accumulator ACC is placed in the accumulator ACC (5), and the system returns to step (3). If the answer is NO, the contents of the accumulator ACC are written to the start block number area (SB) 33 of the directory block 30 (6). Next, data is written to the data area of the block indicated by the accumulator ACC. At this point, it is ascertained whether the 235-byte capacity of the block indicated by the accumulator ACC has been exceeded (8). If the answer is YES, the chain block area CB of the block indicated by the accumulator ACC is placed in the accumulator BCC (9). Next, the write count WCNT for the block indicated by

the accumulator BCC is incremented by one (10). At this point, it is ascertained whether the write count WCNT exceeds 10,000 (11). If the answer is YES, the chain block area CB of the block indicated by the accumulator BCC is placed in memory (12) and [the system] returns to step (10). If the answer is NO, the contents of the accumulator BCC are written to the chain block area CB of the block indicated by the accumulator ACC, and [the system] returns to step (7).

If, on the other hand, the determination in step (8) is NO, the chain block area CB of the block indicated by the accumulator ACC is written to the unused start block number OSB (14). Next, the write count WCNT of the pointer block 1a is incremented by one (15). Next, "FF_{16}" is written to the chain block area CB of the block indicated by the accumulator ACC (16). The contents of the accumulator ACC are then written to the end block number area 34 of the new file location in the directory block 30 (17). Next, the write count WCNT of the directory block 30 is updated (18).

Merits of the Invention

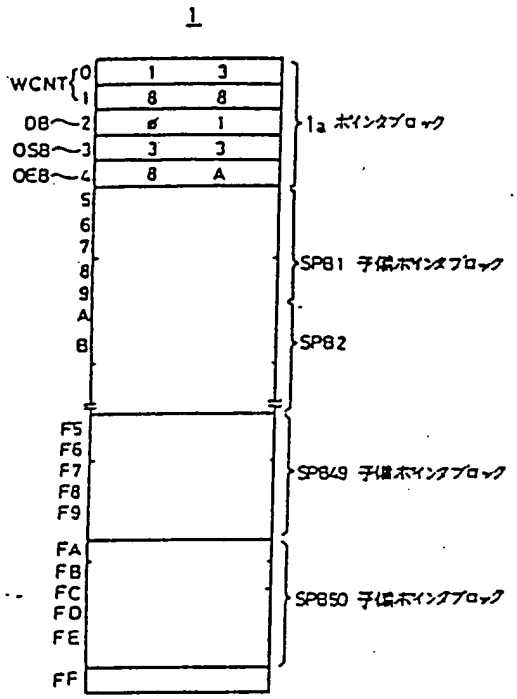
As described above, the present invention divides the memory area into a plurality of blocks, stores in memory the number of writes to each block, and disables write operations to a block when a preset number of writes has been exceeded. [Inadvertent] erasure of data that has been written to the EEPROM can be prevented, and blocks that are no longer used are attached to the end of the unused block, allowing the number of times that unused blocks are used to be equalized.

4. Brief Description of the Figures

Fig. 1(a) is a schematic illustration of a system for managing the number of writes to a programmable read-only memory in a practical example of the present invention. Fig. 1(b) is a block diagram illustrating an example of device configuration in this invention. Fig. 2 is a schematic illustration of the configuration of the EEPROM depicted in Fig. 1(a). Fig. 3 is a schematic illustration of the configuration of the directory blocks depicted in Fig. 2. Fig. 4 is a schematic illustration of the EEPROM prior to use. Figs. 5(a) and (b) are schematic representations of the write operation to the EEPROM. Fig. 6 is a flow chart describing the control procedure for data write operations to the EEPROM depicted in Fig. 1(a).

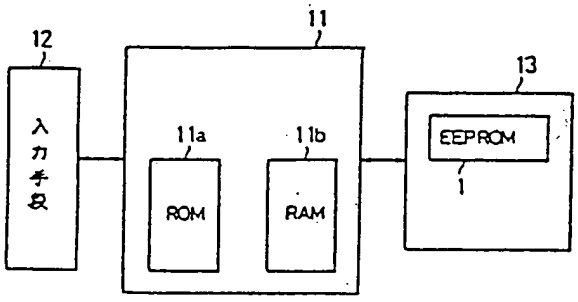
In the drawings, 1 indicates an EEPROM, 1a indicates a pointer block, 21 indicates a block number, 30 indicates a directory block, 31 indicates an update counter, 32 indicates a file area, 33 indicates a start block number area, 34 indicates an end block number area, and 35 indicates a chain block area.

Fig. 1(a)



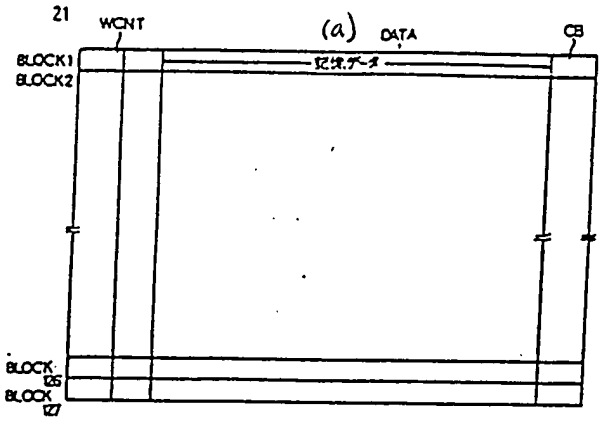
[1a-pointer block; SPB1-spare pointer block; SPB49-spare pointer block; SPB50-spare pointer block]

Fig. 1(b)



[12-input means]

Fig. 2



[(a)-stored data]

Fig. 3

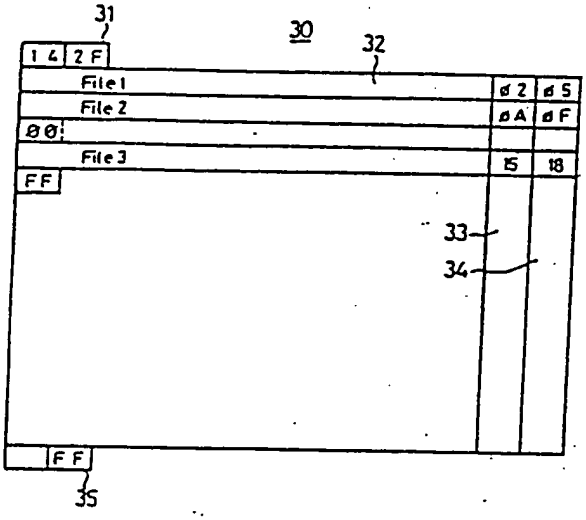


Fig. 4

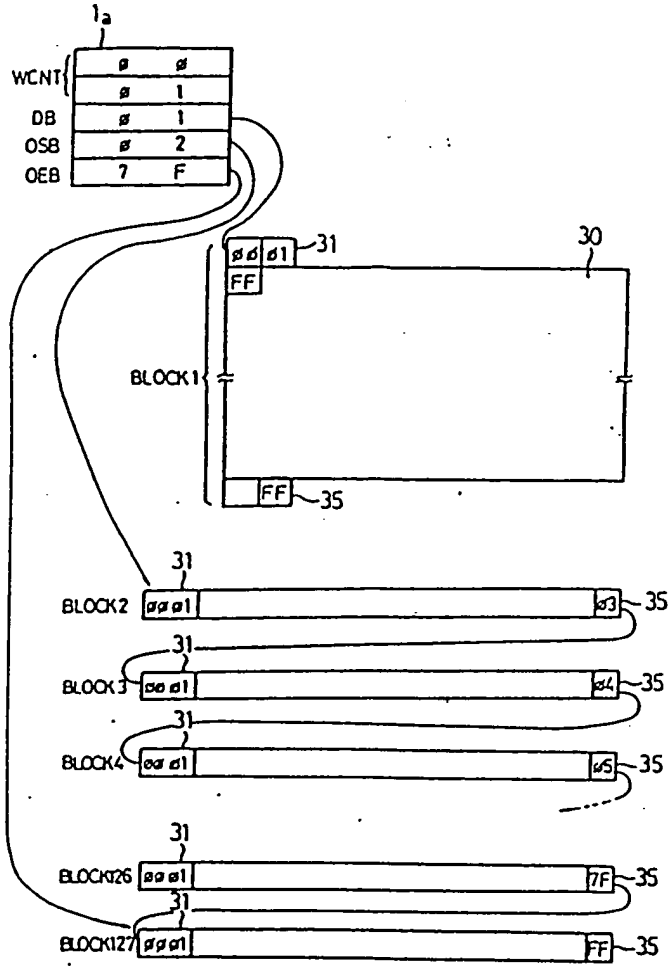

1a
WCNT
DB
OSB
OEB
30
31
BLOCK1
35
BLOCK2
BLOCK3
BLOCK4
BLOCK126
BLOCK127

Fig. 5

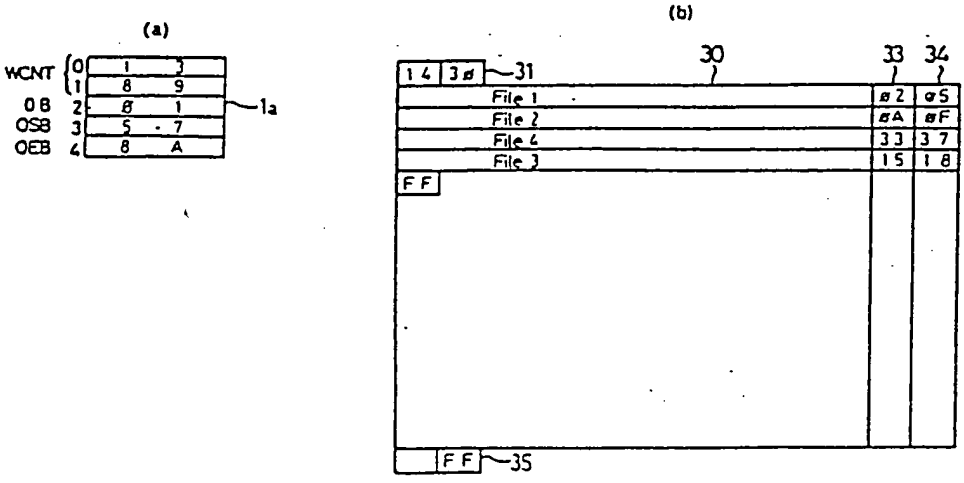
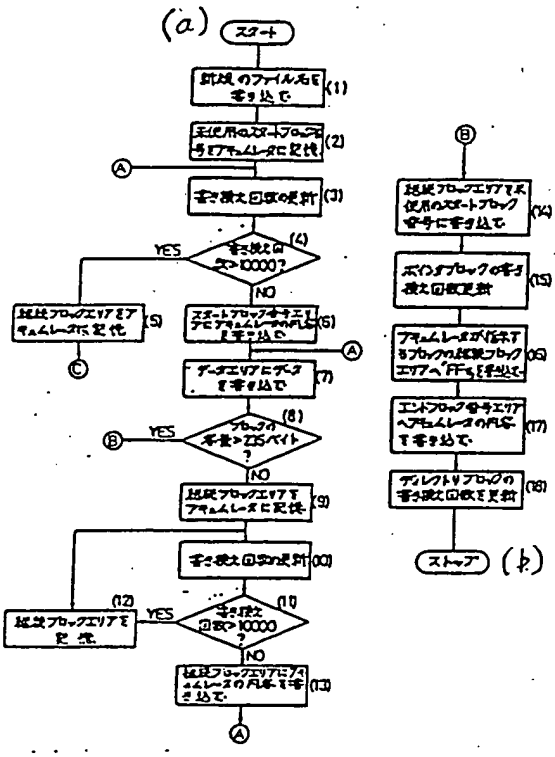

(a)
WCNT
DB
OSB
OEB
1a
(b)
30
31
33
34
File 1
File 2
File 4
File 3
35

Fig. 6



[(a)-Start; (1)-Write new file name; (2)-Store unused start block number in accumulator; (3)-Update write count; (4)-Write count > 10,000?; (5)-Store chain block area in accumulator; (6)-Write accumulator contents to start block number area; (7)-Write data to data area; (8)-Block contents > 235 bytes?; (9)-Store chain block area in accumulator; (10)-Update write count; (11)-Write count > 10,000?; (12)-Place chain block area in memory; (13)-Write accumulator contents to chain block area; (14)-Write chain block area to unused start block number; (15)-Update write count in pointer block (16)-Write "FF_{16}" to chain block area of block indicated by accumulator; (17)-Write accumulator contents to end block number area; (18)-Update write count in directory block; (b)-Stop]

(54) MANAGEMENT SYSTEM FOR OF NUMBER OF TIMES OF WRITING PROGRAMMABLE READ ONLY MEMORY

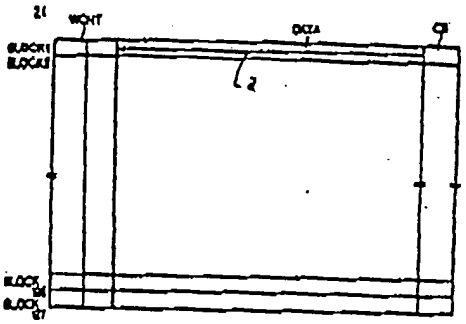
(11) 62-283497 (A) (43) 9.12.1987 (19) JP
(21) Appl. No. 61-124732 (22) 31.5.1986

(71) CANON INC (72) SHINICHI NAKADA

(51) Int. Cl⁴. G11C17/00

PURPOSE: To average the rewriting frequency of a memory block and to prolong the life of an EEPROM by suppressing the writing to the memory block reaching the number of times of setting.

CONSTITUTION: The memory area of the EEPROM in which the erasing, the rewriting or the like are carried out by an input means, a CPU or the like is divided into 127 such as blocks BLOCK1~BLOCK127 and in the respective blocks, an area WCNT for storing the updating and rewriting number of times as well as a data memory area DATA is provided. When the contents of the area WCNT in which the counted value of the updating counter of the directory area of a block pointer is written are referred to and reach the set number, the rewriting of the block is suppressed through the CPU. The rewriting frequency of the respective blocks is averaged and the life of the EEPROM is prolonged.



a: storage data

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　昭62－283497

⑤Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　昭和62年(1987)12月9日
G 11 C 17/00	307	6549－5B	

審査請求　未請求　発明の数 1　（全8頁）

㊹発明の名称　プログラマブルリードオンリメモリの書き込み回数管理方式

㉑特　願　昭61－124732

㉒出　願　昭61(1986)5月31日

㉗発明者　仲田　眞一　東京都大田区下丸子3丁目30番2号　キヤノン株式会社内

㉑出願人　キヤノン株式会社　東京都大田区下丸子3丁目30番2号

㉔代理人　弁理士　小林　将高

明　細　書

1．発明の名称

プログラマブルリードオンリメモリの書き込み回数管理方式

2．特許請求の範囲

記憶領域に書き込まれた情報を電気的に消去可能なプログラマブルリードオンリメモリにおいて、前記記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させることを特徴とするプログラマブルリードオンリメモリの書き込み回数管理方式。

3．発明の詳細な説明

〔産業上の利用分野〕

この発明は、電気的消去可能なプログラマブルリードオンリメモリの書き込み回数の管理方式に関するものである。

〔従来の技術〕

従来のＥＥＰＲＯＭ（Electrical Eresable and Programmable ROM）は、容量も少なく、また書き込むために必要な外部回路が多かった。さらに、チップ内のすべてのデータを消去するモードしか有していなかった。最近は、容量も大きくなるとともに、外部回路も殆ど必要なくＣＰＵのアドレスバス，データバスに結線できるようになり、またＥＥＰＲＯＭ内の1バイトのデータのみの消去も可能となってきた。以上の改良により、使用目的によっては、従来のランダムアクセスメモリ（ＲＡＭ）で構成していた機能の置換が可能となった。

例えば、従来の小型パソコン、日本語ワープロで作成したプログラムや文章，外字等を保存しておくためにメモリカードと云うものがある。これは、必要なときにパソコン，日本語ワープロ等の本体に差し込んでプログラムや文章を記憶させ、本体から引き抜いても、そのデータを記憶しているように、メモリカード内にはＲＡＭと電池が搭載されていた。そこで、メモリカードをＥＥＰＲＯＭで構成することにより、電池を無くすること

ができると考えられた。

〔発明が解決しようとする問題点〕

ところが、ＥＥＰＲＯＭでは従来のＲＡＭのように自由に何度も書き換えられない制約があり、すなわち、あらかじめ設定される書き込み回数を越えて、メモリカードへの書き込みを行なうことにより、記憶しているはずのデータを消失させてしまう等の問題点があった。またＥＥＰＲＯＭに書き込まれたデータのうち、頻繁に書き換えられるデータと書き換え頻度の少ないデータとが存在し、書き換え頻度の高いデータの書き換え回数が所定値を越えると、ＥＥＰＲＯＭへの書き換えが可能にも関わらず書き換え不能となる問題点があった。

この発明は、上記の問題点を解消するためになされたもので、ＥＥＰＲＯＭに書き込まれるデータの消失を防止するとともに、ＥＥＰＲＯＭへの書き込み回数を平均化させるとともに、ＥＥＰＲＯＭ上の書き換え頻度を平均化して、ＥＥＰＲＯＭへの書き換え寿命を延命できるプログラマブルリードオンリメモリの書き込み回数管理方式を得ることを目的とする。

〔問題点を解決するための手段〕

この発明に係るプログラマブルリードオンリメモリの書き込み回数管理方式は、記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させる。

〔作用〕

この発明においては、記憶領域を各ブロック毎に書き込み回数を記憶しておき、この書き込み回数があらかじめ設定される書き込み回数を越えたら、そのブロックへの書き込みを抑止させる。

〔実施例〕

第１図（ａ）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図であり、１はＥＥＰＲＯＭで、例えば書き込み容量が32768 バイト×８ビットで、書き込み回数が１万回に設定してある。ＥＥＰＲＯＭ１は、ポインタブロック１ａおよび予備ポインタブロックＳＰＢ１～ＳＰＢ５０より構成される。ポインタブロック１ａは４アドレス（各１バイト）で構成され、『０～１』番地の２バイトで、書き換え回数ＷＣＮＴ、例えば『１３８８$_{16}$』を記憶している。またポインタブロック１ａの『２』番地の１バイトは、ディレクトリＤＢ、例えば『０１$_{16}$』を記憶している。さらに、ポインタブロック１ａの『３』番地の１バイトは、未使用のスタートブロック番号ＯＳＢ、例えば『３３$_{16}$』を記憶している。またポインタブロック１ａの『４』番地の１バイトは、未使用のエンドブロック番号ＯＥＢ、例えば『８Ａ$_{16}$』を記憶している。

第１図（ｂ）はこの発明の装置構成の一例を説明するブロック図であり、１１はＣＰＵで、ＲＯＭ１１ａ、ＲＡＭ１１ｂを有し、ＲＯＭ１１ａに格納された第６図に示すフローに準じたプログラムに応じて各部を制御する。１２は入力手段で、データ書き込み装置１３にセットされるＥＥＰＲＯＭ１へのデータ書き込みおよびデータ消去を指示する。なお、ＣＰＵ１１にはデータの転送を行うアキュムレータＡＣＣ，ＢＣＣを有している。

第２図は第１図（ａ）に示すＥＥＰＲＯＭ１の構造を示す模式図であり、２１はブロック番号であり、例えば１２７個のブロックＢＬＯＣＫ１～ＢＬＯＣＫ１２７に分割されている。各ブロックは、例えば２５６バイトで構成され、先頭の２バイトで、そのブロックが更新された回数、すなわち、後述する更新回数が記憶されている。次に続く２５３バイトは記憶データＤＡＴＡが記憶されており、最後の１バイトは、記憶データＤＡＴＡがこのブロックに留まるか、または他のブロックに及ぶかどうかを示す継続ブロックエリアＣＢがあり、他のブロックに記憶データＤＡＴＡが及ぶ場合は、継続ブロックエリアＣＢには継続するブロック番号が記憶され、他のブロックに記憶データＤＡＴＡが及ばない場合は、継続ブロックエリアＣＢには『ＦＦ$_{16}$』が記憶されている。

第３図は第２図に示す各ディレクトリブロック構造を説明する模式図であり、３０は前記ディレ

クトリＤＢに指示されるディレクトリブロック、３１は前記ディレクトリブロック３０の更新カウンタで、例えば２バイトで構成される。３２はファイル領域で、各ファイル名が１２バイトで記憶される。３３はスタートブロック番号エリア（ＳＢ）で、例えば１バイトで構成され、ファイルのスタートブロック番号が記憶されている。３４はエンドブロック番号エリア（ＥＢ）で、例えば１バイトで構成され、ファイルのエンドブロック番号が記憶されている。３５はチェーンブロックエリア（ＣＢ）で、ディレクトリブロック３０に継続するディレクトリブロックの有無を記憶する。例えばチェーンブロックエリア３５が『ＦＦ$_{16}$』となる。なお、ディレクトリブロック３０は、例えば１８個のファイル領域３２で構成される。

次に第１図（ａ）および第３図を参照しながらＥＥＰＲＯＭ１の構造について説明する。

第１図（ａ）に示すようにポインタブロック１ａの書き換え回数ＷＣＮＴに、例えば『１３８$_{16}$』が記憶されているとすると、５０００回の更新が行われたことを示し、またディレクトリＤＢには『０１$_{16}$』が記憶されているので、ディレクトリＤＢに指示されるディレクトリブロック３０のブロック番号が『１』で、そのディレクトリブロック３０の更新カウンタ３１には、『１４２Ｆ$_{16}$』が記憶されている。これは、このディレクトリブロック３０を５１６７回更新したことを示し、ファイル領域３２のファイル（ＦＩＬｅ）１（ファイル名）はスタートブロック番号エリア３３が『０２$_{16}$』で、エンドブロック番号エリア３４が『０５$_{16}$』となっているため、ブロックＢＬＯＣＫ２から始まり、ブロックＢＬＯＣＫ５で終ることになる。またファイル領域３２のファイル２は、スタートブロック番号エリア３３が『０Ａ$_{16}$』で、エンドブロック番号エリア３４が『０Ｆ$_{16}$』となっているため、ブロックＢＬＯＣＫ１０から始まり、ブロックＢＬＯＣＫ１５で終ることになる。さらに、ファイル領域３２のファイル３（ファイル名）は、スタートブロック番号エリア３３が『１５$_{16}$』で、エンドブロック番号エリア３４が『１８$_{16}$』となっているため、ブロックＢＬＯＣＫ２１から始まり、ブロックＢＬＯＣＫ２４で終ることになる。またファイル領域３２のファイル３の次に『ＦＦ$_{16}$』が書かれているので、このファイル領域３２はファイル３で終了していることになる。

第４図は未使用のＥＥＰＲＯＭ１の状態を説明する模式図であり、第１図（ａ）、第３図と同一のものには同じ符号を付している。

この図から分かるように、未使用のＥＥＰＲＯＭ１のポインタブロック１ａの書き換え回数ＷＣＮＴが『０００１$_{16}$』、ディレクトリＤＢが『０１$_{16}$』、未使用のスタートブロック番号ＯＳＢが『０２$_{16}$』、未使用のエンドブロック番号ＯＥＢが『７Ａ$_{16}$』がそれぞれポインタブロック１ａの０番地から４番地にそれぞれ記憶されている。これにより、ディレクトリＤＢに指示されるブロックＢＬＯＣＫ１を参照すると、更新カウンタ３１に『０００１$_{16}$』が書き込まれているとともに、ファイル領域３２のファイル１に『ＦＦ$_{16}$』が書き込まれており、さらに、チェーンブロックエリア３５に『ＦＦ$_{16}$』が書き込まれており、ＥＥＰＲＯＭ１が未使用状態であることを示している。

さらに、ポインタブロック１ａのスタートブロック番号ＯＳＢおよびエンドブロック番号ＯＥＢには『０２$_{16}$』、『７Ｆ$_{16}$』がそれぞれ書き込まれている。すなわち、ブロックＢＬＯＣＫ２～１２７には先頭の２バイトに『０００１$_{16}$』が書き込まれ、最終の１バイトに各後続のブロックの継続を示すチェーンブロックエリア３５には、ブロックＢＬＯＣＫ２～１２６に対して『０３～７Ｆ$_{16}$』が書き込まれ、ブロックＢＬＯＣＫ１２７のチェーンブロックエリア３５には『ＦＦ』が書き込まれている。このように、各ブロックＢＬＯＣＫ２～１２７は１つのチェイン構造となる。

次に第３図、第５図（ａ）、（ｂ）を参照しながらＥＥＰＲＯＭ１への書き込み動作を説明する。

第５図（ａ）、（ｂ）はＥＥＰＲＯＭ１への書き込み動作を説明する模式図であり、第１図

（a），第3図と同一のものには同じ符号を付している。なお、書き込み直前は、第3図に示す状態であったものとする。

まず、各ブロックＢＬＯＣＫのファイル領域３２の先頭が『００$_{16}$』のところを探し当てる。第3図の場合は、ファイル2とファイル3との間に『００$_{16}$』があり、そこにファイル4という名前を１２バイトで書き込み、ポインタブロック1aの未使用ブロックのスタートブロック番号ＯＳＢを参照して、スタートブロック番号ＯＳＢの指示するブロックＢＬＯＣＫ、すなわち『５７$_{16}$』の先頭の2バイト情報、すなわち、更新カウンタ３１を『１』インクリメントし、その加算値が、例えば1万回を越えているようであれば、ファイル4のチェーンブロックエリア３５が示すブロックＢＬＯＣＫに対して同様の操作を行い、更新カウンタ３１が1万回以下のブロックＢＬＯＣＫを探し当てて、そのブロックＢＬＯＣＫの番号をポインタブロック1aのスタートブロック番号ＯＳＢに書き込むとともに、ファイル4のデータをブロックＢＬＯＣＫ８７（２５３バイト）に書き込み、ブロックＢＬＯＣＫ８７に越れるようであれば、ブロックＢＬＯＣＫ８７のチェーンブロックエリア３５の指示するブロックＢＬＯＣＫの更新カウンタ３１を『１』インクリメントして加算値が、例えば1万回を越えているかどうかを調べ、指示されるブロックＢＬＯＣＫの更新カウンタ３１が1万回を越えるようであれば、更新回数が1万回以下のブロックＢＬＯＣＫを探し当て、そのブロックＢＬＯＣＫの番号を直前に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に書き込む。このようにして、データの書き込みが行われ、更新回数が1万回を越えるブロックＢＬＯＣＫが排除されて行く。そして、書き込みデータがなくなるまで同様の操作を行い、最後に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に記憶されていた内容を新しい未使用のスタートブロック番号ＯＳＢに書き換え、ポインタブロック1aの書き換え回数ＷＣＮＴを『１』インクリメントして『１３８９$_{16}$』となり、最後にデータを書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５を『ＦＦ$_{16}$』にする。そして、ディレクトリブロック３０の最終ブロック番号を記憶するエンドブロック番号エリア３４に最後のデータを書き込んだブロックＢＬＯＣＫの番号を書き込むとともに、更新カウンタ３１を『１』インクリメントすると、第5図（b）に示されるように、更新カウンタ３１が『１４３０$_{16}$』となり、ファイル4のスタートブロック番号エリア３３が『３３$_{16}$』で、エンドブロック番号エリア３４が『３７$_{16}$』となる。

次に第5図（a），（b）を参照しながらＥＥＰＲＯＭ1に書き込まれているファイル1の削除動作について説明する。

ディレクトリブロック３０となるブロックＢＬＯＣＫ1よりファイル1を探し、ファイル領域３２の先頭の2バイトを『００$_{16}$』とする。次いで、ディレクトリブロック３０の更新カウンタ３１を『１』インクリメントし、ファイル1のスタートブロック番号エリア３３とエンドブロック番号エリア３４のデータを参照して、ポインタブロック1aのエンドブロックＯＥＢが指示するブロックのチェーンブロックエリア３５の内容（削除直前までは『ＦＦ$_{16}$』であった）をスタートブロック番号エリア３３の内容に変更し、このブロックの更新カウンタ３１を『１』インクリメントする。すなわち、未使用ブロックの最後に今削除したファイル4を接続するわけである。このようにして、更新カウンタ３１を進めながら何度もファイルの更新，削除を実行して行くうちに、更新カウンタ３１が1万回に接近する。

次に更新カウンタ３１が1万回に到達した場合のアクセス処理について説明する。

まず、ポインタブロック1aのスタートブロック番号ＯＳＢの内容が示しているブロックＢＬＯＣＫのチェーンブロックエリア３５の内容を新規のスタートブロック番号ＯＳＢとする。次いで、このブロック直前のディレクトリブロック３０の更新カウンタ３１の情報以外の内容を転送する。そして、ポインタブロック1aのディレクトリＤ

Bに新規のディレクトリブロック番号を書き込み、ポインタブロック1aの書き換え回数WCNTおよび更新カウンタ31を『1』インクリメントする。

一方、ポインタブロック1aの書き換え回数WCNTは1万回を越えた場合は、予備ポインタブロックSPB1～SPB50のうち一番近い予備ポインタブロックへ書き換え回数WCNTの情報以外のデータを転送し、新規のポインタブロックの書き換え回数WCNT（0000_{16}）を『1』インクリメントして『0001_{16}』に設定する。この場合、破棄されたポインタブロック1aの書き換え回数WCNTは1万回以上となり、新のポインタブロック1aの書き換え回数WCNTは1万回以下となる。このようにして、カウンタブロック30およびポインタブロック1aの書き込み削除を管理する。また削除されたファイルが使用していたブロックは未使用ブロックの一番最後に回される。これは、未使用ブロックの使用回数を平均化するためである。

第6図は第1図（a）に示したEEPROM1のデータ書き込み初期動作を説明するためのフローチャートである。なお、(1)～(18)は各ステップを示す。

まず、ディレクトリブロック30の空エリアを探して、新規のファイル名を書き込む(1)。次いで、未使用のスタートブロック番号OSBをCPU11のアキュムレータACCに記憶させる(2)。アキュムレータACCが指示するブロックの書き換え回数WCNTを＋1更新する(3)。ここで、書き換え回数WCNTが10000を越えたかどうかを判断し(4)、YESならばアキュムレータACCの指示するブロックの継続ブロックエリアCBをアキュムレータACCに記憶し(5)、ステップ(3)に戻り、NOならばディレクトリブロック30のスタートブロック番号エリア（SB）33にアキュムレータACCの内容を書き込む(6)。次いで、アキュムレータACCが指示するブロックのデータエリアにデータを書き込む(7)。ここで、書き込みデータがアキュムレータACCが指示するブロックの容量が235バイトを越えるかどうかを判断し(8)、YESならばアキュムレータACCが指示するブロックの継続ブロックエリアCBをアキュムレータBCCに記憶させる(9)。次いで、アキュムレータBCCが指示するブロックの書き換え回数WCNTを＋1更新する(10)。次いで、書き換え回数WCNTが10000を越えたかどうかを判断し(11)、YESならばアキュムレータBCCの指示するブロックの継続ブロックエリアCBを記憶させ(12)、ステップ(10)に戻り、NOならばアキュムレータACCが指示するブロックの継続ブロックエリアCBにアキュムレータBCCの内容を書き込み(13)、ステップ(7)に戻る。

一方、ステップ(8)の判断でNOの場合は、アキュムレータACCが指示する継続ブロックエリアCBを未使用のスタートブロック番号OSBに書き込む(14)。次いで、ポインタブロック1aの書き換え回数WCNTを＋1更新する(15)。次いで、アキュムレータACCが指示するブロックの継続ブロックエリアCBへ「FF_{16}」を書き込む(16)。そして、ディレクトリブロック30の新ファイル位置のエンドブロック番号エリア34へアキュムレータACCの内容を書き込む(17)。次いで、ディレクトリブロック30の書き換え回数WCNTを更新する(18)。

（発明の効果）

以上説明したように、この発明は記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、あらかじめ設定される書き込み回数を越えたブロックへの書き込みを抑止させるようにしたので、EEPROMに書き込まれるデータの消失を未然に防げるとともに、不要になったブロックを未使用ブロックの最後尾に接続するようにしたので、各ブロックの書き込み回数を平均化できる利点を有する。

4．図面の簡単な説明

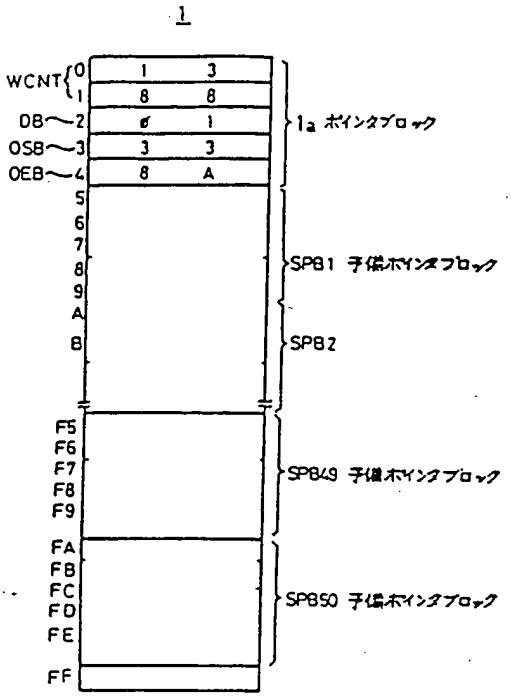
第1図（a）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図、第1図（b）はこの

発明の装置構成の一例を説明するブロック図、第２図は第１図（ａ）に示すＥＥＰＲＯＭの構造を示す模式図、第３図は第２図に示す各ディレクトリブロック構造を説明する模式図、第４図は未使用のＥＥＰＲＯＭ状態を説明する模式図、第５図（ａ），（ｂ）はＥＥＰＲＯＭへの書き込み動作を説明する模式図、第６図は第１図（ａ）に示したＥＥＰＲＯＭのデータ書き込み動作を説明するためのフローチャートである。

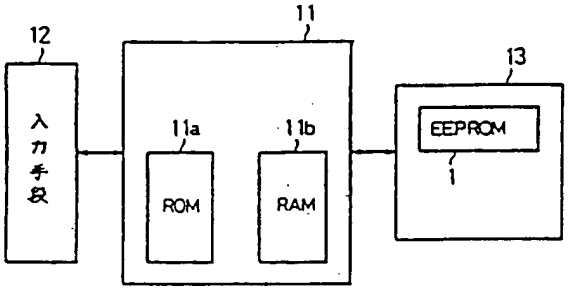
図中、１はＥＥＰＲＯＭ、１ａはポインタブロック、２１はブロック番号、３０はディレクトリブロック、３１は更新カウンタ、３２はファイル領域、３３はスタートブロック番号エリア、３４はエンドブロック番号エリア、３５はチェーンブロックエリアである。

代理人　小　林　将　高

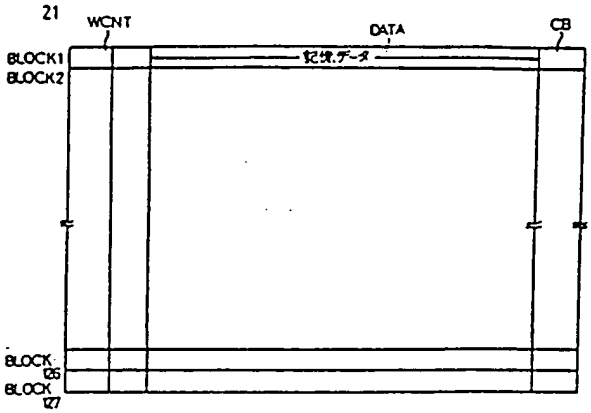
第　１　図　(a)



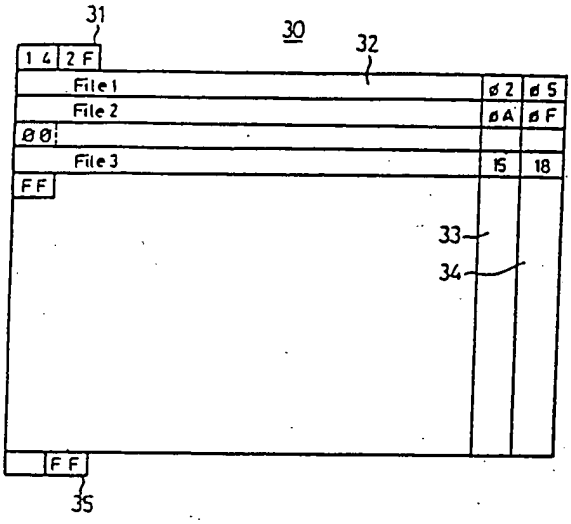
第　１　図　(b)



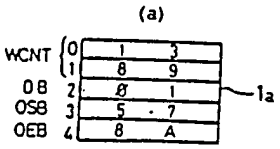
第　２　図

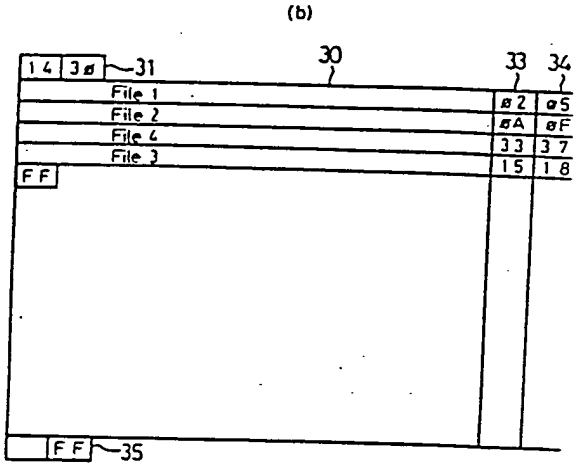


第 3 図

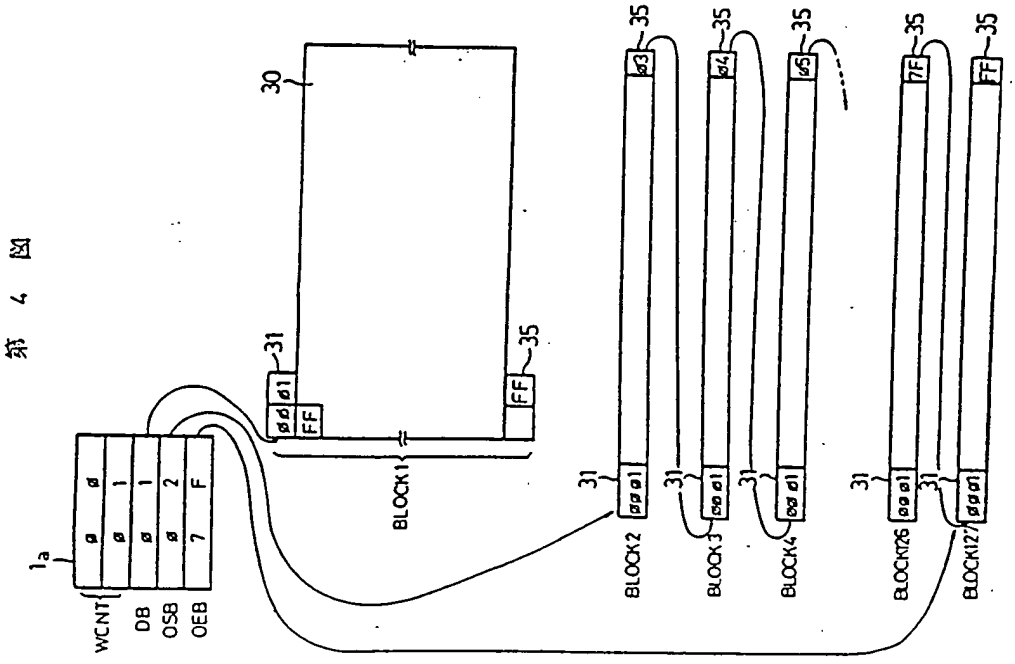

31
30
32
14
2F
File 1
File 2
00
File 3
FF
02
05
0A
0F
15
18
33
34
FF
35

第 5 図

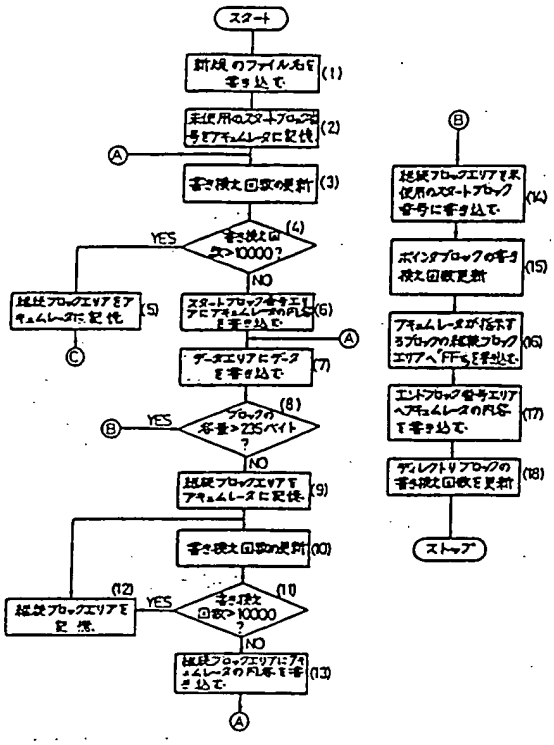

(a)
WCNT
0
1
2
3
4
DB
OSB
OEB
1
3
8
9
0
1
5
7
8
A
1a


(b)
14
30
31
30
33
34
File 1
File 2
File 4
File 3
02
05
0A
0F
33
37
15
18
FF
FF
35

第 4 図


30
31
00
01
FF
35
FF
BLOCK1
1a
WCNT
DB
OSB
OEB
BLOCK2
BLOCK3
BLOCK4
BLOCK126
BLOCK127
03
04
05
7F
FF

第 6 図


スタート
新規のファイル名を書き込む (1)
未使用のスタートブロック番号をアキュムレータに記憶 (2)
書き換え回数の更新 (3)
書き換え回数>10000? (4)
YES
NO
現状ブロックエリアをアキュムレータに記憶 (5)
スタートブロック番号エリアにアキュムレータの内容を書き込む (6)
データエリアにデータを書き込む (7)
ブロックの容量>235バイト? (8)
YES
NO
現状ブロックエリアをアキュムレータに記憶 (9)
書き換え回数の更新 (10)
書き換え回数>10000? (11)
YES
NO
現状ブロックエリアを記憶 (12)
現状ブロックエリアにアキュムレータの内容を書き込む (13)
現状ブロックエリアを未使用のスタートブロック番号に書き込む (14)
ポインタブロックの書き換え回数更新 (15)
アキュムレータが指示するブロックの現状ブロックエリアへ"FF"を書き込む (16)
エントブロック番号エリアへアキュムレータの内容を書き込む (17)
ディレクトリブロックの書き換え回数を更新 (18)
ストップ